Re: Streaming XML and SAX

[Lists Home | Date Index | Thread Index]

- *From*: Nathan Kurz <nat @vall yt l.n t>
- *To*: xml-dev@ic.ac.uk (xml-dev list)
- *Date*: Sat, 27 Feb 1999 16:37:38 -0600 (CST)

David Megginson writes:
> No, it still looks like a messy architecture to me, because the
> transport layer has to know about the packets -- it has to parse the
> XML about to get information about what it's looking at, and that adds
> complexity and inefficiency. A clean architecture should separate the
> layers completely, and use XML only where it has an obvious advantage
> over other approaches.
>
> Tom Harding replies:
> > It's amazing how two people can see things so differently. I think
> > it's supremely elegant that only the XML processor needs to look at
> > data coming off the wire. It's also as efficient as it gets.
>
> David Megginson counters:
> It is efficient only if you know for certain that you need to use a
> single object model for all of the XML information that you're
> receiving; otherwise, you'll end up building a generic object model
> (like a DOM), then tearing it down to build an optimised
> domain-specific one (such as a vector graphic or a
> financial-transaction object), and that process would be painful.

Building a DOM everytime is inefficient, but I have to agree with Tom that having XML act as the protocol as well is quite elegant. Why presume that the XML processor capable of handling the protocol layer would have to build a _generic_ object model? And why presume that an XML processor has to build a _single_ object from all the information?

```
>   <purchase xmlns="http://www.ecommerce.net/ns/ec/">
>    <seqno>12345678</seqno>
>    <customer-id>87654321</customer-id>
>    <vendor-id>18273645</vendor-id>
>    <invoice-id>81726354</invoice-id>
>    <total>92674.12</total>
>   </purchase>
```

It seems like parsers could be made a whole lot more configurable than they currently are. If more configurable, the top level XML processor could build the domain-specific objects itself. Continuing with your <purchase> example, I can envision a processing model like this:

```
Parser sees:   <purchase>

Checks: Is a 'purchase' parser registered?

     Yes: Pass control to it, 'purchase' parser reads until
          </purchase>, then returns control to top level parser.
  or Yes: Slurp text until </purchase>, pass "<purchase>...</purchase>"
          (unparsed) to a 'purchase' parser running under another thread
  or Yes: Slurp text until </purchase> and store it (unparsed) in the
          DOM to be handled on a later pass.
```

```
    No: keep parsing text and adding nodes to the DOM.
or  No: Throw away text (unparsed) up until </purchase>
```

It would then be up to the subparser to build its own objects which could be used later. Or the subparser could return an already processed node to be inserted into the generic object model (or DOM). Is this model possible with any existing parsers?

> Parsing is relatively easy (though it's wasteful to do it twice);
> building an object model from the parsing is time- and
> resource-consuming.

Building the object model is probably the more expensive part, but in many cases multiple selective parsing passes (skimming) would be more efficient than parsing everything completely the first time through. It seems that all current parsers assume that their duty is always to create a faithful model of all of the entire document they are presented with, and thus parse the entire document in a single pass with a single thread of control. Why this assumption?

nathan kurz
nate@valleytel.net

- **Follow-Ups:**
  - **RE: Streaming XML and SAX**
    - *From:* "Didier PH Martin" <martind@netfolder.com>

- Prev by Date: **Re: Streaming XML and SAX**
- Next by Date: **Re: Streaming XML and SAX**
- Previous by thread: **Re: Streaming XML and SAX**
- Next by thread: **RE: Streaming XML and SAX**
- Index(es):
  - **Date**
  - **Thread**
